Laboratorio 2010

Principios de Programación

Introducción

Este documento presenta el problema que deberá resolverse para la aprobación del laboratorio del curso 2010.

Se presenta información acerca de: normas, recursos, plazos, una presentación general del problema, las especificaciones del mismo, ejemplos de ejecución y la forma de entrega.

Información general .

El estudiante que no respete alguna de las consideraciones que siguen corre el riesgo de que su trabajo sea invalidado, con la consiguiente pérdida del curso.

Compilador

Todos los programas deben ser compatibles con el compilador del curso (). No se aceptarán como válidas aquellas tareas que pudieran funcionar con algún otro compilador distinto GCC 3.4.2 y GDB 5.2.1, pero no funcionen con esté. El SO ha utilizar será Windows XP.

Sitio para descargar el dev c++ con el compilador del curso:
http://sourceforge.net/projects/dev-cpp/files/Binaries/Dev-C%2B%2B%204.9.9.2/devcpp-4.9.9.2_setup.exe/download

Sitio oficial de dev c++:
http://www.bloodshed.net/dev/devcpp.html

Individualidad

Esta segunda tarea se debe realizar de manera individual o en grupos de 2 personas. Los grupos se deberán registrar el día de la presentación del laboratorio. Los estudiantes que no estén registrados algún grupo deberán realizar la tarea en forma individual. No podrá haber intercambio de estudiantes entre distintos grupos en las distintas entregas. Los estudiantes permanecerán en su grupo hasta el final del laboratorio.
Para todas las tareas rige el Reglamento del Instituto de Computación de Facultad de Ingeniería(Udelar) ante Instancias de No Individualidad en los Laboratorios. A continuación se adjunta un fragmento del mismo; ante cualquier duda se recomienda leer el documento completo

(http://www.fing.edu.uy/inco/cursos/prog1/pm/field.php/Laboratorio/NoIndividualidad)

Los laboratorios deben ser realizados únicamente por los integrantes del grupo establecido La realización de los laboratorios es estrictamente individual, sea a nivel unipersonal en el primer caso, o del grupo establecido en el segundo. Se entiende que compartir total o parcialmente cualquier actividad del laboratorio atenta contra la integridad del estudiante universitario y de su formación, y por lo tanto constituye una falta grave. Específicamente no es posible compartir por ninguna vía entre integrantes de grupos distintos las tareas de codificación, digitación, compilación, depuración y documentación de los programas u objetos (o entregas) del laboratorio. Además de que no se pueden compartir actividades del laboratorio, no se pueden compartir los productos de las mismas. Cada grupo es responsable de su trabajo de laboratorio y de que el mismo sea individual, independientemente de las causas que pudiesen originar la no individualidad. A modo de ejemplo y sin ser exhaustivos: utilización de código realizado en cursos anteriores (por otros estudiantes) u otros cursos, perder el código, olvidarse del código en lugares accesibles a otros estudiantes, prestar el código o dejar que el mismo sea copiado por otros estudiantes, dejar la terminal con el usuario abierto al retirarse, enviarse código por mail, utilizar código suministrado por terceros, etc. Asimismo se prohíbe el envío de código al grupo de noticias del curso, dado que el mismo será considerado como una forma de compartir código y será sancionado de la manera más severa posible.

Descripción del Juego Batalla Naval.

La batalla naval es un conocido juego, donde dos jugadores disponen de dos tableros cada uno. En uno de los tableros el jugador dispone en las celdas de este sus distintos tipos de barcos, que el otro jugador deberá tratar de ir encontrando en cada turno del juego. En el otro tablero será usado por el jugador para ir registrando las zonas del mapa del jugador contrario que vaya descubriendo en cada turno(en cada ataque). Cuando un jugador descubra todos los barcos enemigos, será el ganador del juego.

¿Que hay que implementar?

Al comenzar cada partida, cada jugador debe primero que nada debe ubicar sus barcos en su mapa. Para esto cada jugador debe disponer de un archivo de estrategias creado previamente, que con ayuda de librerías brindadas por los docentes, podrá cargar en memoria en una estructura de datos adecuada para su utilización (esto se describe más abajo en este documento). Luego de que cada jugador dispuso todos sus barcos (hay distintos tipos de barcos que ocupan distinta cantidad de celdas, todas ellas consecutivas horizontalmente o verticalmente).Para este laboratorio habrá tres tipos de barcos, nos referiremos a ellos cómo barcos de tipo A, tipo B, y tipo C. Tanto los barcos de tipo A y B ocuparan dos celdas consecutivas, y los de tipo C tres. Utilizaremos los símbolos ‘A’,’B’,’C’ para identificar estos tipos de barcos en el tablero. Es decir, por ejemplo si un barco de tipo a esta ubicado en el mapa en las coordenadas (0,1) y (0,0), en el tablero en esas posiciones se encontrara el símbolo ‘a’. El juego se desarrolla por turnos. En el momento que cada jugador dispone de su turno, elije una posición en el mapa enemigo, que no haya sido seleccionada antes para atacarlo. En el momento del ataque puede ocurrir que no de a ningún blanco (llamémosle a esta situación “agua”), o que acierte a una sección de un barco enemigo. Si un barco enemigo es atacado completamente (son descubiertas todas las celdas que este ocupa), el programa debe notificar al jugador que hizo el ataque, que ha hundido completamente un barco. El juego finaliza cuando uno

de los dos jugadores haya hundido todos los barcos enemigos. Este jugador será el ganador de la partida.

Nuestro tablero tendrá una cantidad de filas y de columnas igual a **TAM_TABLERO,** tal que 4< **TAM_TABLERO** <10.Las filas y columnas se enumeran empezando desde cero hasta **TAM_TABLERO-1.**

Estructuras Principales.

Nota: Todas estas estructuras y enumerados se encuentran declarados en el archivo útil.hpp

Las estructuras de datos que representan y se utilizan en el juego son la siguientes:

```
typedef enum estado_juego{TURNO_JUGADOR1='1',
TURNO_JUGADOR2='2',GANO_JUGADOR1='3',GANO_JUGADOR2='4' };

typedef enum tipo_celda {OCULTA='*', AGUA='|', BUQUE1='A', BUQUE2='B',
PORTA_AVIONES='C'};

typedef struct Jugador{
    tipo_celda  tableroPropio[TAM_TABLERO][TAM_TABLERO]
    ,  tableroEnemigo[TAM_TABLERO][TAM_TABLERO];

};
```

Información General del la segunda entrega del laboratorio

Se pide

En la segunda entrega el estudiante deberá implementar de forma completa el juego de la batalla naval. El juego a implementar deberá soportar que dos jugadores humanos puedan jugar entre ellos.El programa al ejecutarse imprimirá en pantalla un menú inicial donde se mostraran las siguientes opciones:

1-Jugar

2-Mostrar Top Diez

3-Salir

El usuario deberá ingresar en consola uno de los valores entre 1 y 3.

Nota: Para todo el laboratorio, asuma que todas las entradas ingresadas por el teclado son válidas.

1. **Si se oprime la tecla 1**, comienza una nueva partida y se deberá pedir que se ingresen los nombres de los dos jugadores, y la ruta absoluta del archivo de estrategias de cada uno de ellos.(El archivo de estrategias se describe posteriormente).

   Ejemplo:

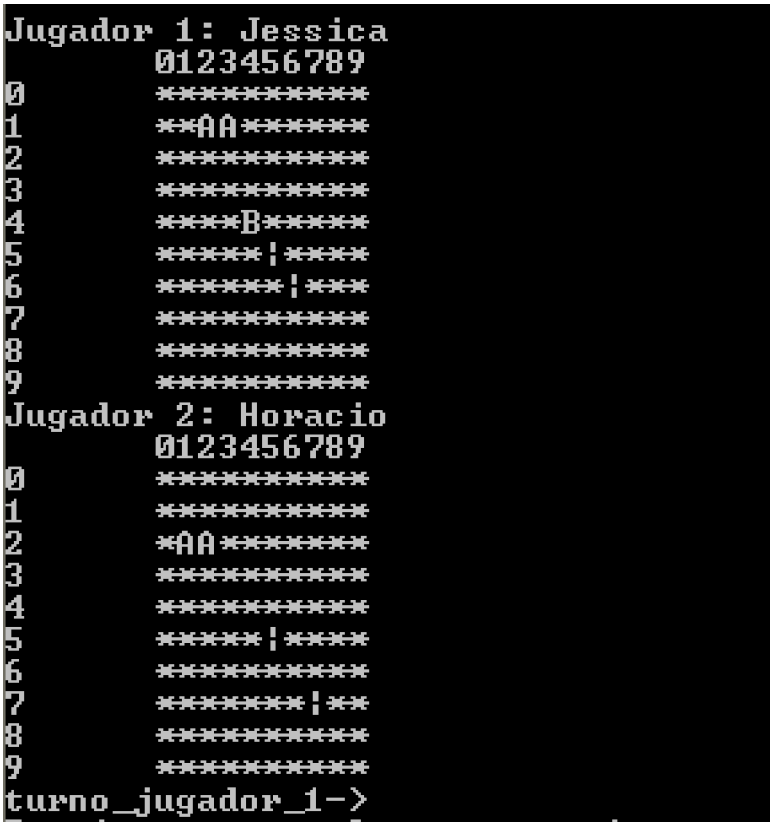
   ->Ingrese el nombre del Jugador 1: Jessica

   à Ingrese la ruta del archivo que contiene el mapa del jugador 1: c:\jesy.txt

   ->Ingrese el nombre del Jugador 2: Horacio

   à Ingrese la ruta del archivo que contiene el mapa del jugador 2: c:\archivo2.txt

   Luego de obtenidos estos datos y cargado los archivos (para cargar los archivos -que contienen el tablero- se utilizan las funciones proporcionadas por los docentes que se encuentran definidas en el archivo util.hpp), comenzará el juego. Se mostrara en todo momento en pantalla el contenido del tablero de los dos jugadores, junto con el nombre de cada uno de ellos, para mostrar el estado del juego, de la siguiente forma:



En pantalla no deberán aparecer más de dos tableros, por lo que será necesario entre dos jugadas limpiar la pantalla. Debajo del segundo tablero, se deberá mostrar un Promp.(turno_jugador_X->). Donde se muestre cual es el jugador que tiene el turno asignado para atacar (la X puede ser 1 o 2).

El jugador que tenga el turno asignado, puede disparar al jugador enemigo indicando la fila, y la columna que desea atacar del tablero enemigo (la fila y la columna están separados por un '-').

Ejemplo: turno_jugador_1->2-2, indica que el jugador 1 quiere atacar la coordenada(2,2), del tablero enemigo.

Al disparar a un enemigo pueden ocurrir los siguientes escenariosr:

- Que se ataque a una posición que ya se encuentre descubierta, por lo que el turno seguirá perteneciendo al jugador atacante, pero se mostrara un mensaje de advertencia indicando lo ocurrido. El mensaje es: "La coordenada indicada ya fue atacada."
- Que la coordinada ingresada nunca antes haya sido atacada, por lo que se revelara el símbolo correspondiente al elemento escondido en esa posición (agua, barco de tipo A,B, o C).En caso de que en el último ataque se haya completado el bombardeo a todos los barcos enemigos. La partida finalizara, mostrando el mensaje "Gano el jugador X".(la X se corresponde al numero de jugador que gano-1 o 2-).Luego de esto se retornará al menú principal. Se deberá guardar en memoria los nombres de los jugadores y la cantidad de ataques que hayan realizado en una determinada partida, de los 10 últimos mejores jugadores. La forma de

comparación de la “performance” de cada jugador, será midiendo la cantidad de ataques que tuvo que realizar para ganar la partida, es decir el mejor jugador será aquel que le llevo realizo menor cantidad de ataques a su enemigo para ganar.

2. **Si se oprime la tecla 2**, el programa mostrara en orden decreciente los 10 mejores jugadores, junto con la cantidad de ataques realizados.(En caso de haber menos partida se mostraran solamente las que hayan).El formato de salida en pantalla para un ejemplo cualquiera podría ser:

Jugador	**Cantidad de Ataques**
1-Pedro	**7**
2-Santiago	**15**
3-Maria	**22**
4-Juan	**23**
5-Marta	**32**

¡¡¡Presione una tecla para volver al menú!!!

Luego de mostrar en pantalla, se pedirá la opción de apretar una tecla y cuando se oprima se volverá al menú principal.

3. **Si se oprime la tecla 3** el programa finalizará imprimiendo en pantalla el mensaje “**Adios-TecnoInf 2010**”.

Forma de entrega

Se indicara por medio del docente responsable las forma de entrega.

Fecha de Entrega

8/06/10

Consideraciones Generales:

Se puede utilizar todo lo visto en las clases teóricas y prácticas.

En esta tarea como en todos los problemas de este curso, se valorará además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera, se hará énfasis en buenas prácticas de programación que lleven a un código legible, bien documentado y mantenible, tales como:

- indentación adecuada
- utilización correcta y apropiada de las estructuras de control
- modularidad del código utilizando procedimientos y funciones
- no repetición de código
- código claro y legible
- algoritmos razonablemente eficientes
- utilización de comentarios que documenten y complementen el código
- utilización de constantes simbólicas
- nombres mnemotécnicos para variables, constantes, funciones, etcétera.

Anexo 1 Implementación

A la hora de implementar la tara, podrá hacer uso de los procedimientos y funciones implementados en la primera entrega. Se deberán utilizar las estructuras de datos definidas en la primera sección de este documento. No se pueden alterar las estructuras dadas.

Se brindara al estudiante el archivo “util.hpp”, donde se dispone de las funciones necesarias para cargar los tableros con la información contenida en los archivos de estrategia.

Anexo 2 Archivos de Estrategía

Los archivos de estrategia son archivos donde cada jugador debe indicar en que coordenadas quiere ubicar sus barcos. El formato del archivo es el siguiente:

Debe contener 7 líneas, cada una de ellas salvo al última deben contener un salto de línea (enter).

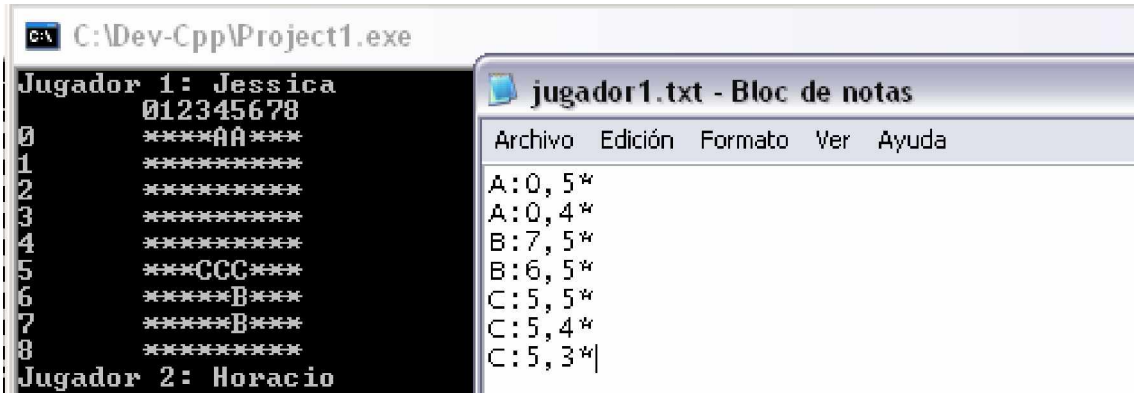
Las líneas deben estar formadas de la siguiente forma,

x:fila,columna*

Donde x puede tomar valores incluidos en {A,B,C}, fila y columna son dos naturales entre 0 y TAM_TABLERO-1.

Recordar que un barco de tipo A y uno de tipo B ocupan dos celdas, por lo que ocuparan dos líneas cada uno de ellos en este archivo, y un barco de tipo C ocupa tres celdas, por lo que habrá tres líneas en este archivo para indicar la posición del barco.

Ej:



Recordar que la ultima línea no debe tener un ’enter’.

Anexo 3 Ejemplo de utilización de útil.hpp para cargar un archivo

```
….
#include "util.hpp"
….


…
int main{
    ….
    tipo_celda tablero [TAM_TABLERO][TAM_TABLERO];
    tipo_celda tablero2 [TAM_TABLERO][TAM_TABLERO];


   inicializarTablero(tablero, OCULTA);
   inicializarTablero(tablero2, OCULTA);
   cargaTablero("c:\\jugador1.txt",tablero);
   cargaTablero("c:\\jugador2.txt",tablero2);
……
}
```

Nota: tener en cuenta que es importante inicializar el Tablero antes de cargarlo.
